

SOFTWARE QUALITY



LESSON 5: WEB TROJANS

MALWARE

Malicious Software

- Viruses



- Trojans



- Bots



- ...

- Things that bad guys use to compromise your system ...

- Steal your resources

- Steal your personal stuff

- ...

TYPES OF MALWARE

Not all types of Malware are destructive.

- But they may cause very **annoying** behaviors.
 - Generate bunch of ads
 - Cause your computer run slowly

These types of programs are not classified as Malware, by some experts.

- They are known as **Potentially Unwanted Programs (PUP)** or **Potentially Unwanted Applications (PUA)**.
- Sometimes, Bundled in legitimate software as a package, but
 - Negatively affect a computer
 - Introduce other security risks
 - Buy something that you do not need

Adware is software that displays unwanted **advertising** on a computer or mobile device.

- usually in the form of pop-up ads
- redirect your browser to a certain website

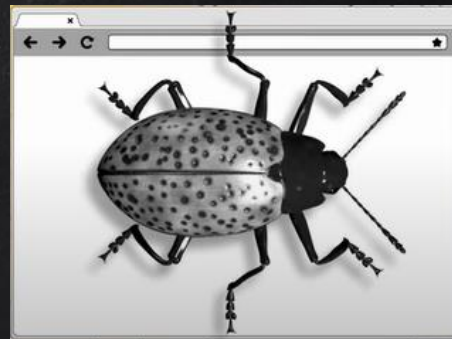
- While it usually does not cause any direct harm to the user's device, but it can be very annoying behaviour
 - can sometimes contain **spyware**.

BROWSER HIJACKER

Hijacking a browser means that the malicious software has **redirected** your computer's browser to a **different website**, generally used to display advertising.

It can be used to:

- **generate visits** to a certain website.
- lead you to a **malicious website** that will **download malware** onto your computer.



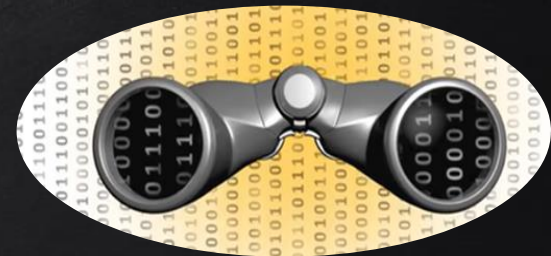
SPYWARE

It is designed to **spy**.

It hides on your computer and **monitors** everything you do.

It can:

- track the **web activity**
- access **email**
- **steal** your username and password



RANSOMWARE

With ransomware, someone can **lock up** your computer

- holding it hostage and forcing you to **pay** a lot of money just to get your files back.
- Regularly **backup** the important things on your computer

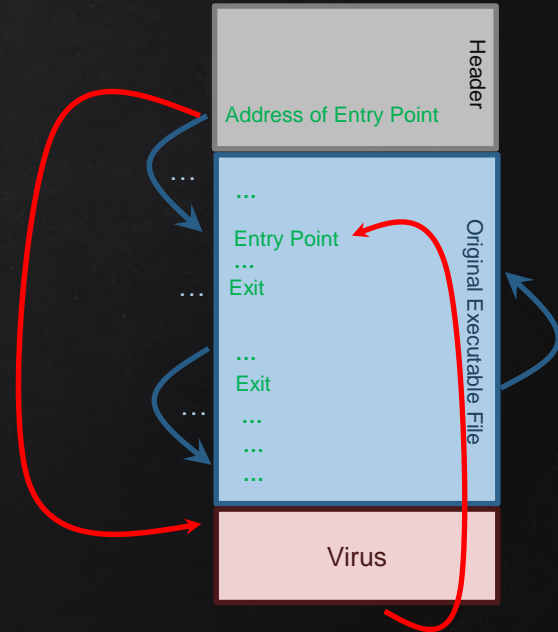
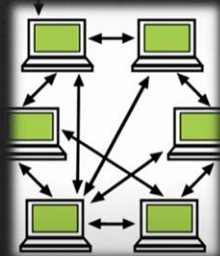


VIRUS

Almost all viruses are attached to an **executable file** which means the virus may exist on a system, but will not spread **until a user runs the infected program**.

Viruses often originate on the internet and spread when:

- downloading a file infected with a virus
- peer-to-peer file sharing
- email attachment
- ...

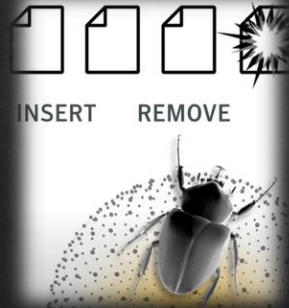


MACRO VIRUS

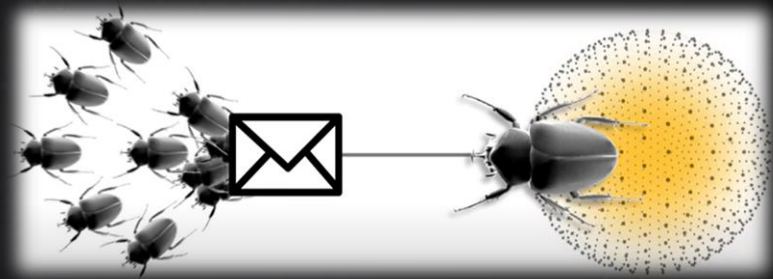
Macro viruses are written specifically to alter **macros** which are common commands that **word processing programs** use.

- Once opened macros can cause changes in the text documents such as

- removing or inserting words
- changing the font
- ...



- some macros can even access email accounts and send out copies of itself to a users contacts.



A computer worm's main objective is to **spread** as many **copies** of itself in any way possible from computer to computer.

a worm can replicate itself without any human interaction.

- it does not need to attach itself to a program in order to cause damage.

Worms can

- modify and delete files
- inject additional malware onto the computer
- ...

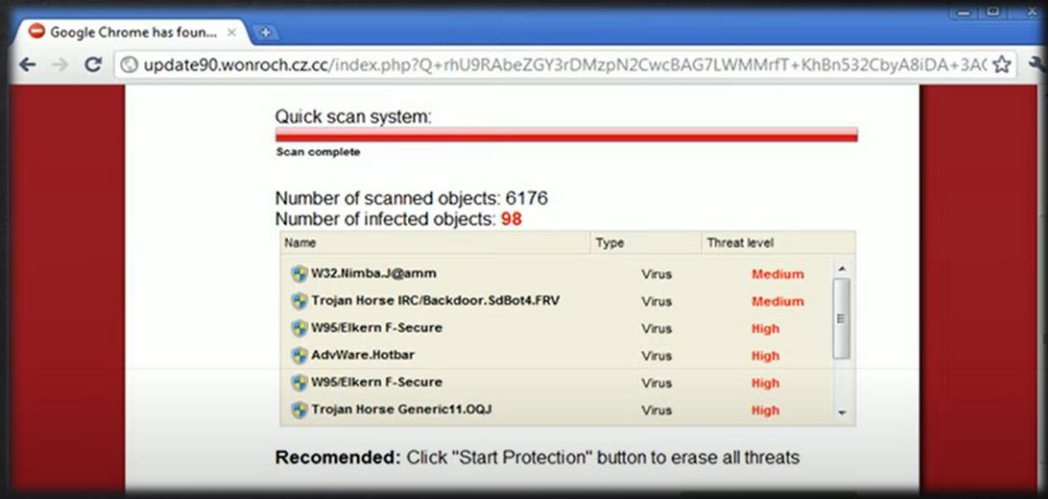


SCAREWARE/SCAMWARE/ROGUEWARE/FAKE ALERT

A malware that says there is a **problem** on your machine and offers to solve it, if you pay them.

Often, it will **pretend to be an antivirus software**, popping up on your screen to tell you that your machine:

- is full of malware
- operating system has errors
- is running slowly
- could crash



COMPUTER TROJAN

WHAT IS A TROJAN?

Trojan horse is something that appears to be a **gift**, but that actually is a **trap**.

In the context of computer security, the term has materialized into “a **program that appears to be cool, but perform some scary damages**, e.g. erases all files.”

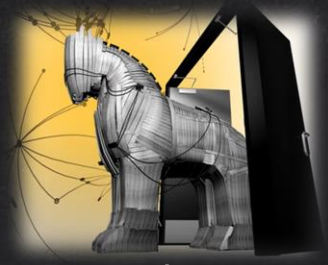


The “Trojan” part comes from the **Greek legend of the Trojan horse**.

TYPES OF TROJANS

Depends of the purpose of the trojan, they can be classified into different types:

- Banking Trojan
- Backdoor Trojan
- Downloader Trojan
- Information Stealer
- Remote Access Trojan
- DDoS Trojan



EXAMPLE 1

Imagine a voting web site allowing users to choose among different alternatives to collect statistics.

```
<form action="http://www.voting.example/vote.asp"
      method="get">
  <input type="radio" name="alt" value="1"/>Foo<br/>
  <input type="radio" name="alt" value="2"/>Bar<br/>
  :
</form>
```

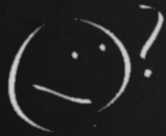
Since the form uses the **GET** method, users voting will visit URLs like this when they submit the form:

```
http://www.voting.example/vote.asp?alt=2
```



HOW IT CAN BE ATTACKED?

16



An attacker can easily **copy** the URL and **mail** it to lots of people and mislead them into the link.

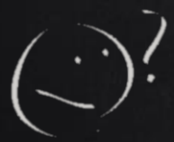
If several of the targeted people follow the link, they all give a vote after the attacker's liking, and the statistics get all **wrong**.

EXAMPLE 2

The really scary threat with Web Trojans is that they work with **authentication** too.

As these trojans function by tricking a user or his browser into visiting a site, things will be done on behalf of the user if he is already **logged in** to the target site.

What happens if the user who **is logged in** to his on-line bank, visit this HTML in the browser:



```
<form name="f" action="https://www.bank.example/pay.asp"
      method="post">
  <input type="hidden" name="from-account"
        value="1234.56.78901"/>
  <input type="hidden" name="to-account"
        value="9876.54.32109"/>
  <input type="hidden" name="amount"
        value="10000.00"/>
</form>
<script>document.f.submit()</script>
```



So, if the victim **is already logged in** to the bank, and the bank accepts that kind of form, the user will unintentionally **transfer** some money to the attacker, or to someone the attacker wants to get trouble into with the authorities.



For this to work, one will have to reach the victim **while he is logged in** to the target site.

- If the user is not always logged in, **the attacker will somehow have to make sure the victim is signed on to the target site** before tricking him into viewing the malicious HTML.
- How? 😊?



- Sometimes that is quite easy:
 - if the user has clicked a “remember me” option, he will always be logged in.
 - if the target site is an intranet solution based on domain authentication.
 - if single sign-on solutions (like Google, Microsoft Passport, etc.) users may be logged in to more sites than they imagine.
- For a site where the attacker may add content, e.g. a discussion forum, that may be easy:
 - just add a very tempting note in the forum asking people to take a look at a link. Anyone reading the note is guaranteed to be logged in:
- Social Engineering to the rescue!



The target site is our bank named www.bank.example

The attacker can send the victim an E-mail that appears to come from the bank:

```
To: victim
From: security@bank.example
Subject: Emergency -- please read immediately!
```

Dear John Doe

Due to recent issues, we kindly ask you to help us check your account. Please immediately log in to the bank. Once logged in, click on this link and follow the instructions:

<http://www.bank.example@520962083/login>

Sincerely,
Cliff Johnson, Chief Security Officer at Example Bank Inc.

PROBLEM

THE PROBLEM

Many web sites, including banks, shops, discussion sites, and whatnot are **vulnerable** to some kind of **Web Trojan** trickery.

- To see how to design a **web solution that is not vulnerable**, we need to understand the problem.

When someone browses our site, we typically generate web pages that contain **URLs** and **forms** inviting the user to **do** something.

- Web Trojans work because it is possible for attackers to give victims these **offers** on our behalf.

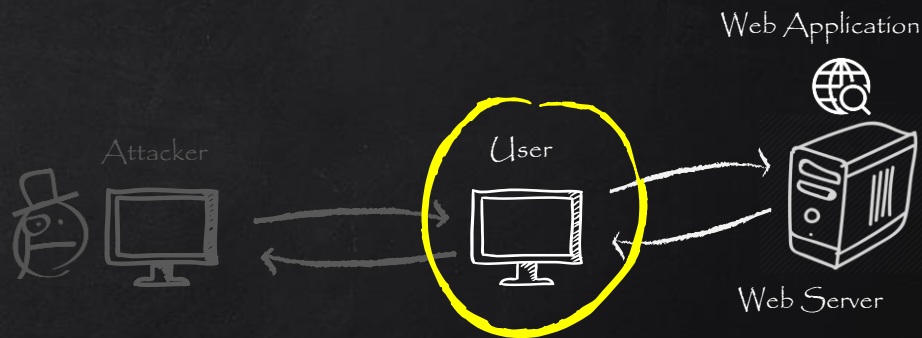
To avoid the threat, we need to make sure **the action** a user takes really is based on **an offer we once gave him**, rather than on an offer given him by someone else.

- How? 😊?



Many developers think that the **Referer** header is a good thing to check to make sure the visitor came from our site.

- In general, the Referer header should not be used for security, as it comes from the **client-side**.



But in the Web Trojan case it could have been useful, **if** it was not for the fact that many **filter it out** for privacy reasons.

- Referer headers are thus often **missing** in totally legitimate requests, so we need to find a method that **does not depend on it**.



An approach that would work would be to require **reauthentication** of the user for **every** action that changes something.

This solution includes adding a **password** field to every form presented to the client—an approach taken by many online banks.



Unfortunately, giving passwords all the time is **cumbersome**, so we will try a different approach.

SOLUTION

A PRACTICAL SOLUTION: TICKET SYSTEM

To protect against Web Trojans, developers should implement a “**ticket system**”.

Central to this system are nonpredictable, random numbers, called **tickets**.

A web page may typically contain **one or more offers** to do an action that has **side effects**.

For each such offer, generate a **unique, random string**, and **connect** it with the offer.

If the offer is a form:

```
<input type="hidden"
      name="ticket" value="uFnVB5oHiMVMcFTN" />
```

If the offer is a link:

```
<a href="vote.jsp?alt=1&ticket=bVGZTa78LV00Zn9n">
  Yes, I agree</a>
```

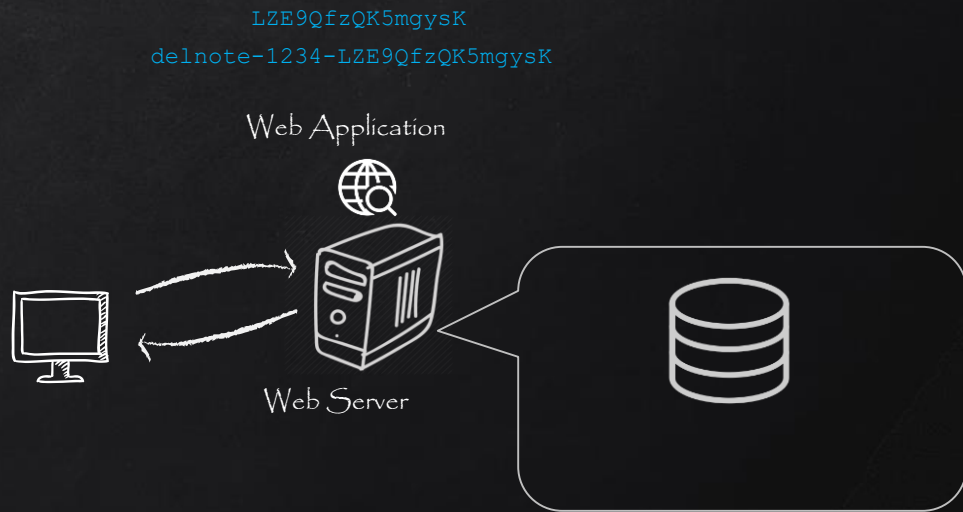


GENERATE THE TICKET

For each ticket generated, add a string naming the action it refers to, and store the combined string in a **ticket pool** in the session of the user who receives the offer.

If the action in question is, for instance, to confirm deletion of a note numbered **1234**, and the ticket is represented as **LZE9QfzQK5mgysK**, the string to store could be **delnote-1234-LZE9QfzQK5mgysK**.

You now have the same ticket on both the client-side and the server side.





PERFORM THE RELEVANT ACTION

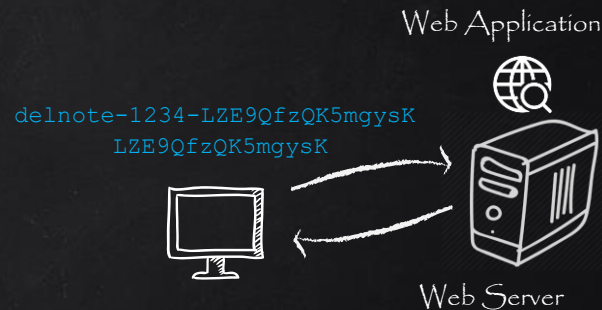
Whenever a request to perform an action arrives from the user, **extract the ticket** from the request.

`delnote-1234-LZE9QfzQK5mgysK`

Then add **the name of the action** that is about to be performed to the beginning of the string, and look for a match in the session ticket pool.

If a **matching** ticket is found, you may assume that the offer was given by your web site.

- In that case, **perform** the action, and **remove** the ticket from the pool.





The ticket system works because attackers **will not be able to guess** what ticket values you may have given to the user, and they will not be able to insert tickets into the victim's ticket pool on the server side.

1. If you include the tickets in **GET requests**, so that they become part of URLs, you risk **ticket leakage** through Referer headers if the user follows a link from your site to other web servers.
2. The system will break if your application is vulnerable to **Cross-site Scripting**. If an attacker is able to insert JavaScript in a page generated by your server, he will be able to extract tickets from the page.

HINTS IN TICKET SYSTEM IMPLEMENTATION

1. Web pages with tickets in them **cannot be cacheable**, as each ticket can only be used once.

Tell browsers and proxies not to cache pages with tickets in them to make sure every page comes with **fresh, valid** tickets.

2. Put an **upper limit** on the number of tickets in a pool.

There will be left-over tickets, as users do not necessarily follow up on all the offers we give them.

When the limit is reached, remove the oldest ticket and add the new one.

The limit will stop people from deliberately filling up memory with unused tickets.



3. Session **timeouts** make the server-side ticket pools disappear. The result is that we may get **legitimate incoming requests** with **no matching ticket** on the server, for instance from a user who has spent an hour filling in lots of details in a web form. Users won't be happy if we **throw away** their input. Instead we could **redisplay** the web page with a **new ticket** and all the incoming text filled in, tagged with an explanatory message that asks him to confirm that he really intends to perform the action.



4. Tickets are needed only for requests that actually **change** something on the server.

A user who wants to edit a note, for example, will first request the note editing form, typically by **clicking a link**.

This request **does not change** anything, so it need not be protected by a ticket.

When he has finished editing, he **POSTs** his changes.

The second request **updates** the server-side database, so it should be **protected by a ticket**.

SUMMARY

- Attackers may give their victims offers on behalf of a target web site and thereby trick them into doing something they never intended to do.
- The offers may be URLs or auto-submitting forms, and they may be given through any channel available, such as E-mails and web pages outside of the target site.
- To protect a web site and its users from these Web Trojans, web developers will need to implement special security mechanisms, for instance the ticket system.

- The ticket system will make sure an action taken is based on **an offer given by the web site** rather than by some off-site attacker.
- As of writing, mechanisms to protect against Web Trojans are not commonly in use, making lots of web sites vulnerable.

YOUR TASKS FOR THIS WEEK



Reading:

- “Innocent Code: A Security Wake-Up Call for Web Programmers (Chapter 5).





Input Validation

- Download Lab **Week 5** Instruction file from course webpage on GitHub.

<https://hogeschool.github.io/Software-Quality/>

Read the Instructions and perform the tasks.